Ingestion and Query System Design for Process Research

# 1. Problem Overview

Researchers need to analyze operating system process data across various OS types.   
The ingestion system must collect, normalize, and store process output data from tools like 'ps' (Linux/macOS) and 'tasklist' (Windows), and allow querying for insights such as the most common applications or average memory usage per process.

# 2. Solution Architecture

The solution is divided into two major parts:  
1. Ingestion Flow: Collects text data and metadata, processes it, and stores both raw and normalized data.  
2. Query Interface: Provides endpoints to query the normalized data for research and reporting purposes.

# 3. System Diagram

The full system architecture is represented in the Lucidchart diagram below:

Lucidchart Link: [lucid chart diagram](https://lucid.app/lucidchart/876126bd-d0fe-4243-8012-8611964df1bb/edit?viewport_loc=-390%2C-3326%2C4606%2C1630%2C0_0&invitationId=inv_fc29207d-17ba-4d65-b6bb-c11cba84dce5)

# 4. System Components

## 4.1 Ingestion API

A FastAPI-based service receives multipart/form-data uploads.   
Each submission includes raw OS command output (text file) and metadata (e.g., OS, machine ID, timestamp).   
The file content is published to a RabbitMQ exchange called 'pre\_normalise' with metadata as headers.

## 4.2 RabbitMQ Message Queue

Used for decoupling ingestion from processing. Raw text files are published to the 'pre\_normalise' exchange and consumed by a raw input logger and a normalization service.

## 4.3 Raw Data Consumer

A lightweight service that listens on the 'raw\_process\_input' queue and stores the raw file + metadata into a PostgreSQL database under the 'raw\_process\_data' table.

## 4.4 Normalizer

A service that parses the command output (e.g., from 'ps auxww' or 'tasklist'), extracts structured process records, and stores them in 'normalized\_process\_data'.   
Optional: extra format-specific fields can be saved in a JSONB column or in format-specific tables (e.g., 'extra\_ps\_data').

## 4.5 PostgreSQL Database

Stores both raw and normalized data. TimescaleDB can be added as an extension to support time-series optimization for trend queries and data retention policies.

## 4.6 Query Interface

Provides endpoints for researchers to run queries like:  
- Top 10 most common processes on Windows machines  
- Average memory usage of 'chrome.exe' on macOS  
- All processes run on a given machine during a date range

# 5. Summary

This modular, scalable design ensures ingestion can handle data from various OS tools, normalization logic can evolve independently, and the data remains accessible for research via a robust querying interface.  
It balances structure (via PostgreSQL + optional TimescaleDB) with flexibility (using JSONB or optional extra tables).